二体力学系における相互作用のシミュレーションモデル

著者	旭 貴朗
著者別名	Asahi Takao
雑誌名	経営論集
号	80
ページ	31-38
発行年	2012-11
URL	http://id.nii.ac.jp/1060/00004502/

二体力学系における相互作用のシミュレーションモデル

Simulation Model of Interaction in Two-Body Dynamical Systems

旭　　貴朗

1　はじめに

本稿では、二体力学系に対するシミュレーションモデルを、モデル理論的アプローチ[1]に従って言語 CAST で作成し[2]、シミュレーション開発実行環境 Simcast で実行する[3]。対象となる二体力学系とは、二つの物体が引力によって引き合って運動しているシステムのことである。この系内の物体の運動の軌跡（軌道）を求めることは「二体問題」と呼ばれている。二体問題はすでに物理学では解析的に解かれている[4]が、本稿ではモデル理論アプローチの立場から改めてシミュレーションモデルを作成し、軌跡を求めることとする。

物理学では、二体問題を一体問題に帰着させ、一体問題を解いている。そこでは、まず二体力学系の変数を変換することによって、換算質量をもつ一つの物体の運動方程式を導出し、その軌跡を完全に求める。そして変数を逆変換することにより各物体の軌跡を導出するのである。これにより、方程式の解として、任意の時刻 t における各物体 (i=1,2) の位置を厳密に定める関数 $u_i(t)$を得ることになる。

しかし、ここで行なうシミュレーションはそれとは事情が異なることに注意したい。一般にシミュレーションでは、各物体の位置 $u_i(t)$が与えられたとき、微小な時間 h が経過した後の位置 $u_i(t+h)$ を（近似）計算し、これを繰り返しながら、先々の時刻における位置を計算するのである。各時刻の位置 $u_i(t)$, $u_i(t+h)$, $u_i(t+2h)$…をグラフ上に描画していけば、結果として運動の軌跡を得ることができる。本稿でも、このような方針でモデルを作成する。

また、モデル理論アプローチの立場から二体力学系を考察する理由は、それが相互作用をもつシステムの例であるからである。この系では常に二つの物体の間に相互に引力が働いている。周知のように、引力は質量の積に比例し、距離の 2 乗に反比例するが、ニュートン力学の範囲内では質量は変化しないので、位置が重要な状態変化の要因である。さらに、物体 1 の位置が変化すると同時に相手の物体 2 の位置も変化する。したがって物体間の距離だけでなく引力の働く方向も変化していく。ここでも位置が重要な変化の要因である。引力は加速度を生み、加速度は速度を変化させ、その結果として位置が変化していく。これが物体間の相互作用の正体である。

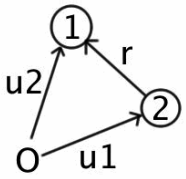
私の研究の大きな目的は、対象システムがどのような性質をもつとき、どのようにシミュレーションモデルを作成していくのか、そのモデル構造や開発方法をシステム理論の観点から明らかにすることである。この観点から言えば、「常に相互作用をしている二つのオブジェクトから成るシステム」は最も基本的な考察対象である。したがって本稿における二体力学系は具体的な例としてふさわしいものと考えられる。つまり、本稿の目的は、相互作用をもつ基本的なシステムの具体例として二体力学系のシミュレーションモデルを作成することにより、そのモデル構造の特性を明らかにすることである。

以下、第2節では、物理学にもとづく二体力学系の数理モデルをもとに、近似計算のためのオートマトンモデルを構築する。第3節は、そのオートマトンモデルを言語CASTで記述したユーザモデルを提示する。第4節は実行結果である。第5節で、本稿の主題である相互作用について考察し、シミュレーションモデルの構造の一般的な特性を明らかにする。第6節は結語である。

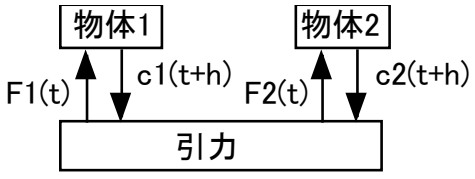
2　二体力学系のオートマトンモデル

二つの物体1, 2が引力によって引き合って運動しているシステムを二体力学系と呼ぶ。以下では太字によりベクトル量を表わすこととする。二つの物体の質量を m_1 と m_2、時刻 t でのそれぞれの位置を $\mathbf{u}_1(t), \mathbf{u}_2(t)$、速度を $\mathbf{v}_1(t), \mathbf{v}_2(t)$、加速度を $\mathbf{a}_1(t), \mathbf{a}_2(t)$ とする。引力 $\mathbf{F}_1(t), \mathbf{F}_2(t)$ は $\mathbf{r}(t) = \mathbf{u}_1(t) - \mathbf{u}_2(t)$ の方向（正または負）に働いている。図表1の記号Oは直交座標系の原点である。

図表1　二体力学系



図表2　二体力学系のモデル構造



さてモデル理論アプローチでは、複数の要素システムが相互作用によって結合した図表2のようなシステムとして、この系をモデル化する。本稿では、各要素システムは状態機械（State Machine）、相互作用は関数としてモデル化する。また全体システムをMoore型オートマトンとしてモデル化する。

各物体の運動をモデル化するにあたり、各物体の時刻 t における「状態」を

$c1(t) = (m_1, \mathbf{u}_1(t), \mathbf{v}_1(t))$

$c2(t) = (m_2, \mathbf{u}_2(t), \mathbf{v}_2(t))$

とする。まず物体1を状態機械<δ1>として定式化する。物体1の運動方程式は $m_1 \cdot \mathbf{a}_1 = \mathbf{F}_1$ であるから、加速度は $\mathbf{a}_1(t) = (1/m_1) \cdot \mathbf{F}_1(t)$ である。近似計算のために、微小時間 h 経過後の速度は $\mathbf{v}_1(t+h) = \mathbf{v}_1(t) + h \cdot \mathbf{a}_1(t)$ と近似し、位置は $\mathbf{u}_1(t+h) = \mathbf{u}_1(t) + h \cdot \mathbf{v}_1(t+h)$ と近似することができる[5]。したがって物体1の「状態遷移関数」は、次

のように論理式で定式化することができる。

$\delta 1(c1(t), \mathbf{F}1) = c1(t+h) \Leftrightarrow$

$\mathbf{a}_1(t) = (1/m_1) \cdot \mathbf{F}_1(t)$,

$\mathbf{v}_1(t+h) = \mathbf{v}_1(t) + h \cdot \mathbf{a}_1(t)$,

$\mathbf{u}_1(t+h) = \mathbf{u}_1(t) + h \cdot \mathbf{v}_1(t+h)$,

$c_1(t+h) = (m_1, \mathbf{u}_1(t+h), \mathbf{v}_1(t+h))$. --- (1)

ここに、記号 ⇔ は必要十分条件を表わす論理記号である。同様にして物体 2 を状態機械$<\delta 2>$として定式化すると式の形は同じであるから、

$\delta 2(c2(t), \mathbf{F}2(t)) = \delta 1(c2(t), \mathbf{F}2(t))$. --- (2)

である。

システム全体の状態は $c(t) = (m_1, \mathbf{u}_1(t), \mathbf{v}_1(t), m_2, \mathbf{u}_2(t), \mathbf{v}_2(t))$ である。このとき、万有引力定数を G、r 方向への単位ベクトルを$(\mathbf{u}_1(t) - \mathbf{u}_2(t))/r$ とすると、万有引力の法則により、物体 1 が物体 2 から受ける力は $\mathbf{F}_1(t) = -(G \cdot m_1 \cdot m_2/r^2) \cdot (\mathbf{u}_1(t) - \mathbf{u}_2(t))/r$ である。また、作用反作用の法則により、物体 2 が物体 1 から受ける力は $-\mathbf{F}_1(t)$ である。ここで、状態 $c(t)$ に対して二物体への力 $(\mathbf{F}_1, \mathbf{F}_2)$ を対応させる関数を、

$\theta(c(t)) = (\mathbf{F}_1(t), \mathbf{F}_2(t)) \Leftrightarrow$

$\mathbf{F}_1(t) = -(G \cdot m_1 \cdot m_2/r^2) \cdot (\mathbf{u}_1(t) - \mathbf{u}_2(t))/r$,

$\mathbf{F}_2(t) = -\mathbf{F}_1(t)$. --- (3)

と定義すると、これが図表 2 における「相互作用」である。

以上、式(1), (2), (3)により、図表 2 を構成する各要素システムが定式化できた。これらを組合せて、全体システムを Moore 型オートマトン$<\delta, \lambda>$として定式化する。システムの「状態」は $c(t) = (m_1, \mathbf{u}_1(t), \mathbf{v}_1(t), m_2, \mathbf{u}_2(t), \mathbf{v}_2(t))$ であることに留意しつつ、全体システムの「状態遷移関数」を次のように定義することができる。

$\delta(c(t)) = c(t+h) \Leftrightarrow$

$(\mathbf{F}_1(t), \mathbf{F}_2(t)) = \theta(c(t))$,

$c1(t+h) = \delta 1(c1(t), \mathbf{F}1)$,

$c2(t+h) = \delta 2(c2(t), \mathbf{F}2)$,

$c(t+h) = (c1(t+h), c2(t+h))$. --- (4)

「出力関数」は二物体の位置を値とする。

$\lambda(c(t)) = b(t) \Leftrightarrow b(t) = (\mathbf{u}_1(t), \mathbf{u}_2(t))$ --- (5)

以上により全体システムがオートマトン$<\delta, \lambda>$として定式化できた。外力はないので、自律的に状態が変化していくオートマトンである。実際のシミュレーションでは、「δ で次の時刻の位置と速度を計算し、λ で二物体の位置（2 点）を描画する」ことを繰り返すことになる。

3　言語 CAST によるモデル

前節のオートマトンモデル$<\delta, \lambda>$を言語 CAST で記述したものが図表 3 である。図表中の preprocess は事前準備を行なう述語である。引力定数 G と時間の刻み幅 h

をグローバル変数で定義し、適当な値を代入した後、グラフウインドウを開き、そのウインドウ番号をグローバル変数 Wp.g に代入している。また、強制的に時間を進めるために、ダミーの"d"を 4000 回入力する（inputsequence()="d"; times()=4000;）[6]。

状態変数は質量と位置と速度を並べたリストである。c=[m1,u1,v1,m2,u2,v2]。初期状態 initialstate ( ) は c0:=[200,[100,100],[0,3],300,[300,100],[0,-2]] とした。

delta1(c,F) = cc は一物体の運動を表す状態遷移関数である（式(1)）。theta(c,F) は相互作用関数である。delta(c, dummy) = cc は全体システムの状態遷移関数である。式(4)を使って次状態 cc:=[u1next,v1next,u2next,v2next] を計算している[7]。最後に、出力関数 lambda(c)=cc では、二物体の位置 [u1,u2] を出力するとき、ついでに述語 xwrite を用いて、すでに開いているウインドウ(Wp.g)に各物体の位置（2 点）を描画する。

図表 3　言語 CAST によるモデル（ユーザモデル）

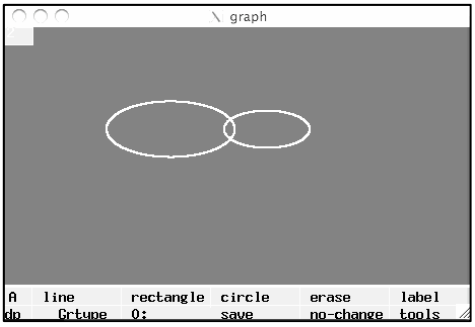
```
//twobody10.set
func([delta1,delta2,theta]);
preprocess() <->
   G.g:=100, h.g:=0.01,
   wopen(Wp,"xyplot"),plaingraph(Wp),Wp.g:=Wp;
inputsequence()="d"; times()=4000;
initialstate()=c0 <->
   c0:=[200,[100,100],[0,3],300,[300,100],[0,-2]];
delta1([m,u,v],F)=cc <->                    //物体 1 の運動
   a:=(1/m)*F,
   v_next:=v + h.g*a,
   u_next:=u + h.g*v_next,
   cc:=[m,u_next,v_next];
delta2(c,F)=delta1(c,F);                    //物体 2 の運動
theta([m1,u1,v1,m2,u2,v2])=[F1,F2] <->      //相互作用
   r:=sqrt(sum((u1-u2)*(u1-u2))),
   F1:= -(G.g*m1*m2/(r*r*r))*(u1-u2),
   F2:= -F1;
delta([m1,u1,v1,m2,u2,v2],dummy)=cc <->     //全体システム
   [F1,F2]:=theta([m1,u1,v1,m2,u2,v2]),
   cc1:=delta1([m1,u1,v1],F1),
   cc2:=delta2([m2,u2,v2],F2),
   cc:=append(cc1,cc2);
lambda([m1,u1,v1,m2,u2,v2])=b <-> b:=[u1,u2],   //出力
   xwrite(Wp.g,"r",append(u1,u1)),
   xwrite(Wp.g,"r",append(u2,u2));
```

4　実行結果

モデル理論アプローチによるシミュレーションでは、テキストエディタで作成したユーザモデルを開発実行環境Simcastに読込んでコンパイルし実行させることができる。図表4は図表3のユーザモデルを実行した様子である。物体1の質量が物体2の質量より軽い（m1＝200, m2＝300）ので左の楕円が大きくなっている。また、初期値をみると、重心の速度はm1・v1＋m2・v2＝3・200 - 2・300＝0であるため、全体の重心の位置は不動である。したがって、二つの楕円がその位置で表示され続けることになる。

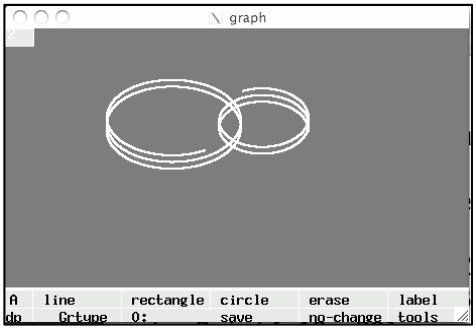
当初は刻み幅h＝0.01が荒いので、楕円を一周してくると曲線がずれて閉じないことを心配したが、この場合には予期に反して閉じた曲線を描くことが分かる。もちろん、近似計算なのでパラメータ値によっては閉じないこともある。質量比を変えた場合も試みた[8]。

図表4　実行の様子



G=100, h=0.01,
c_0=[200,[100,100],[0,3], 300, [300,100],[0,-2]]

図表5　重心が移動していく様子



G=100, h=0.01,
c_0=[200, [100,100],[0,4],300, [300,100],[0,-3]]

図表5は重心の速度がゼロでないように初期速度を変更し、再度実行した結果である。初期値をみると、重心の速度はm1・v1＋m2・v2＝4・200 - 3・300＝-100 である。グラフ内では楕円が上方向に一定の速度でずれていくことが分かる。

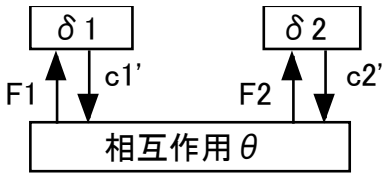
以上、二体力学系に対するシミュレーションモデルをモデル理論アプローチに従って言語CASTで作成し、開発実行環境Simcastで実行した。その結果として、1) 二物体がそれぞれ楕円軌道を描くこと、2) 全体の重心が等速運動を行なうことなどが確認できた。すでに周知の性質であるので新奇性はないが、図表3のモデルの整合性は確認できた。

5　相互作用の構造

さて、本稿の目的はシステム理論の観点から、相互作用をもつシステムのシミュレーションモデルの構造の特徴を明らかにすることである。今回は具体的に二体力学系を対象とし、状態機械<δ1>と<δ2>としてモデル化した要素システムを組み合わせ、

全体システムをオートマトン $<\delta,\lambda>$ としてモデル化した[9]。その構造は図表６のように表わすことができる。要素システム間の影響が相互作用（関数θ）である。

図表６　相互作用をもつシステムの構造



本稿の焦点は要素システム間の相互作用にある。第１項「はじめに」で述べたように二体力学系における相互作用は引力F_iであるが、それは二体間の距離と方向によって定まり、互いの相対的な位置が本質的であった。このことは一つの要素システムの次状態$\delta_i(c_i,F_i)$を定めるには他の要素の状態が必要であることを意味する。本稿では相互作用関数$\theta(c)=(F1,F2)$が各引力を計算している。状態cを参照できることが相互作用関数のポイントである。

恐らく「相互作用が関数で表わせることがシミュレーションの前提条件」であろう。もしも相互作用が関数として表わせるなら、数学的には$\theta(c)=(\theta1(c),\theta2(c))$に分割することができる。ただし$\theta1$は物体１への入力F1を定める関数であり、$\theta2$は物体２への入力F2を定める関数である。状態cを参照しつつも、それぞれの入力はそれぞれに計算することが可能である。したがって自動的に図表６が保証され、シミュレーションモデルの作成が可能となるのである。経験上、相互作用が関数で定式化できることが多いので、「もしも相互作用が関数として表わせるならば、それにより自動的に図表６が保証され、式(4)でモデル化できる」点は重要であろう。

次に、式(4)の全システムの状態遷移関数に、分割された相互作用関数θ_iを代入すると、$\delta(c1,c2)=(\delta1(c1,\theta1(c)),\delta2(c2,\theta2(c)))$と書くことができる。これは制御工学あるいはコンピュータ科学では並列処理と呼ばれるモデルである。同じデータcにもとづいて、$\delta1$と$\delta2$が形式上「同時」に計算される。もしも相互作用がなければ要素システムが自律的に運動できるシステム、すなわち二体力学系のような分散系ならばこれでモデル化できるように見える。

しかし、たとえ自律分散系であっても「衝突」を含む場合には、並列処理モデルがふさわしくない場合がある[10]。そこでは、要素システム１の変化による影響をまず計算し、それをもとに要素システム２の変化による影響を求めている。これは逐次処理のモデルである。対象システムの性質や設計方針にもよるだろうが、逐次処理のモデリングを行なう理由を説明できない。今後の課題である。しかし、少なくとも衝突を含むシステムの場合にはモデリングに注意が必要なことは確かである。

最後に図表６の応用可能性を考える。たとえば各物体上にロケットエンジンを設置し、推進力を制御できるならば、それらの物体からなるシステムは自律分散系である。エンジンをもつ物体は、引力という相互作用を受けながらも、自由に軌道を変えることが可能である。エンジンを制御する主体にとっては、相互作用のもとで（引力はむ

しろ自由行動に対する制約だと感じられるであろうが）物体の軌道を望みの方向に制御したいのである。

その場合には、各物体への入力 F_i にエンジンの推力 F' が加算されることになり、状態遷移関数を $\delta_i(c_i, F_i + F')$ などと変更すればよい。このように図表 6 あるいは式(4)という一般的なモデル構造のなかでは、要素システムのモデル修正が容易であることが想像される。例えば、一体力学系に対する最適制御の問題に対してはモデル理論アプローチによる問題解決システム（Solver）によって求解することができる[11]。二体力学系に対しても対象モデルの修正はわずかであろうと推測できる。

6　おわりに

本稿では、まず、相互作用をもつ基本的なシステムの具体例である二体力学系のシミュレーションモデルを言語 CAST で作成し、モデル理論的アプローチが提供する開発実行環境 Simcast で実行し、モデルの整合性を確認した（第 2 節～第 4 節）。今回は、そのモデルをもとにシステムの相互作用について考察を加えた。その結果、モデル構造の特性から、相互作用が関数で表わせることがシミュレーションの前提条件であろうことが示唆される。もしも相互作用が関数として表わせるならば、それにより自動的に図表 6 が保証され、式(4)の応用可能性が推測できる（第 5 節）。相互作用をもつシステムに対して、モデル理論アプローチによるシミュレーションを作成する上での方針あるいは手がかりの一つとなることが期待できる。

【注】

[1]　モデル理論アプローチの詳細については、高原康彦ほか（2007）『形式手法 モデル理論アプローチ：情報システム開発の基礎』日科技連を参照のこと。

[2]　旭貴朗，高原康彦，中野文平ほか（2008.03）「経営情報システム開発のためのモデル記述言語 CAST」経営情報学会誌，Vol. 16，No. 4，pp.19-30.

[3]　旭貴朗（2009.03）「モデル理論アプローチによるシミュレーション---開発実行環境 Simcast」東洋大学経営論集 73 号，pp. 33-51.

[4]　物理学の基本的な大学教科書を参照のこと。例えば、沢田，影山，小口，柳（1983）『基礎物理 I』（3rd.ed），槇書店，pp. 91−92.

[5]　Yasuhiko Takahara and Yongmei Liu (2006.12), ***Foundations And Applications of MIS : Model Theory Approach,*** Springer-Verlag の第 8 章を参考にした。

[6]　今の所、Simcast ではグラフウインドウに描画できるオブジェクトの数は 2 万個までである。二体力学系では一度の状態遷移で 2 個の点を描画するので、状態遷移（シミュレーションサイクル）は 1 万回が限度である。

[7]　言語 CAST がリスト計算をすることができるので、状態遷移関数を書き直して（システム要素 delta1, delta2, theta を定義することなく）、次のように極端に短く記述することは可能である。

```
delta([m1,u1,v1,m2,u2,v2],a)=cc <->
  r:=sqrt(sum((u1-u2)*(u1-u2))),
  F:=-(G.g*m1*m2/(r*r*r))*(u1-u2),
  [a1,a2]:=[(1/m1)*F,-(1/m2)*F],
```

```
[v1next,v2next]:=[v1,v2] + h.g*[a1,a2],
[u1next,u2next]:=[u1,u2] + h.g*[v1next,v2next],
cc:=[u1next,v1next,u2next,v2next];
```

シミュレーションを実行するだけでいいのなら、これで十分である。しかしながら、本稿では相互作用の考察に焦点があるので、あえて本文第3項のようにシステム要素を陽表的に定式化した。

[8] 質量比 m1/m2 が大きくなると誤差が蓄積され、楕円がずれていくことになる。実際に、m1/m2 = 1000 くらいで楕円がずれることがある。その場合でも刻み幅を小さくすることにより（h = 0.001 程度で）回避できることを確認した。しかしながら、状態遷移の回数に制限があること［注5］に注意する必要もある。場合によっては描画しないで出力 b（位置）をファイルに保存しておき、後で市販ソフトで描画することになろう。出力の保存については、下記 Web サイトの技術資料「ファイル操作」や「スプレッドシートによるデータ表示」が参考になるだろう。

旭貴朗「モデル記述言語 CAST によるシミュレーション」2012.09.01
http://www2.toyo.ac.jp/~asahi/research/simulation/index.html

[9] システム理論では、このようなシステムの全体$<\delta, \lambda>$をプロセスと呼び、その一部である個別の要素システム$<\delta 1>, <\delta 2>$をサブプロセスと呼ぶこともある。

[10] 旭貴朗（2011.11）「モデル理論アプローチによるシミュレーション―自律分散システムの開発手順 ―」『経営論集』78 号, pp.177-188 では、次のようにモデル化している。

$$\delta(c1,c2) = cc \Leftrightarrow$$
$$c1' = \delta 1(c1, \theta 1(c)),$$
$$c2' = \delta 2(c2, \theta 2(c1',c2))$$
$$cc = (c1', c2').$$

これは要素システム 1 の変化による影響 (c1',c2) をまず計算し、それをもとに要素システム 2 の変化による影響 cc を求めている。これは逐次処理のモデルである。

[11] 注[5]の Takahara & Liu (2006) では、一物体の力学系を対象とし最適制御を解いている。ただし、数学的意味の最適解ではなく、いわゆる実用解が得られる。

（2012 年 9 月 10 日受理）